Supplementary Information for Inferring causality in biological oscillators

Jonathan Tyler, Daniel Forger, Jae Kyoung Kim*

*To whom correspondence should be addressed; E-mail: jaekkim@kaist.ac.kr

1 Theoretical Foundations

Here, we describe the theoretical foundations of our main result. In particular, we formally define the reflection time t_Y (Fig. 1B) and prove that, if X positively regulates Y, then the regulation-detection score $\langle R_{X\to Y}\rangle = 1$. Similar arguments can be used for negative regulations.

Definition 1. *If $Y(t)$ is a smooth time series with one maximum (t_M) and one minimum (t_m) over a period, then for $t \neq t_M, t_m$, we define the reflection time t_Y such that $Y(t_Y) = Y(t)$ and $t_Y \neq t$. If $t = t_M, t_m$, then $t_Y := t$.*

Theorem 1. *Let $X(t)$ and $Y(t)$ be smooth time series with period τ that have one maximum and one minimum over a period τ and solve the ODE*

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = f(X, Y). \tag{S1}$$

If f is strictly monotonically increasing in X (i.e., X positively regulates Y), then $R^{t_Y}_{X\to Y}(t) = X^{t_Y}_d(t) \cdot \dot{Y}^{t_Y}_d(t) \geq 0$ for all t, where

$$X^{t_Y}_d(t) = X(t) - X(t_Y)$$
$$\dot{Y}^{t_Y}_d(t) = \dot{Y}(t) - \dot{Y}(t_Y).$$

Moreover, $\langle R_{X\to Y}\rangle = 1$, where

$$\langle R_{X\to Y}\rangle := \frac{\int_0^\tau R^{t_Y}_{X\to Y}(t)\mathrm{dt}}{\int_0^\tau |R^{t_Y}_{X\to Y}(t)|\mathrm{dt}}.$$

Proof. First,

$$\begin{aligned}
R^{t_Y}_{X\to Y}(t) &= X^{t_Y}_d(t)\cdot \dot{Y}^{t_Y}_d(t)\\
&= (X(t)-X(t_Y))(\dot{Y}(t)-\dot{Y}(t_Y))\\
&= (X(t)-X(t_Y))\cdot\\
&\qquad (f(X(t),Y(t))-f(X(t_Y),Y(t_Y)))\geq 0.
\end{aligned}$$

The last inequality holds because, by assumption, $f(X,Y)$ is a monotonically increasing function of X when Y has the same value (i.e., $Y(t)=Y(t_Y)$). Furthermore, $R^{t_Y}_{X\to Y}(t)$ cannot be 0 for all t. For the sake of contradiction, assume that $R^{t_Y}_{X\to Y}(t)=0$ for all t. If $R^{t_Y}_{X\to Y}(t)=X^{t_Y}_d(t)\cdot\dot{Y}^{t_Y}_d(t)=0$, then $X^{t_Y}_d(t)=0$ or $\dot{Y}^{t_Y}_d(t)=f(X(t),Y(t))-f(X(t_Y),Y(t_Y))=0$. If $f(X(t),Y(t))-f(X(t_Y),Y(t_Y))=0$, then $X(t)-X(t_Y)=0$ as well because f is strictly monotonically increasing in X. Therefore, if $R^{t_Y}_{X\to Y}(t)=0$ for all t, then

$$X(t)=X(t_Y)\quad\text{for all } t \tag{S2}$$

Next, from Eqns. (S1) and (S2),

$$\dot{Y}(t)=f(X(t),Y(t))=f(X(t_Y),Y(t_Y))=\dot{Y}(t_Y), \tag{S3}$$

for all t. This implies that $\dot{Y}(t)=\dot{Y}(t_Y)=0$ given the nature of t_Y (e.g, if Y is increasing at t, then Y is decreasing at t_Y). Therefore, $\dot{Y}(t)=0$ for all t, i.e., Y is constant. However, this contradicts the assumption that Y is a smooth oscillating time series. Since $R^{t_Y}_{X\to Y}(t)\geq 0$ and is a nonzero function, $\langle R_{X\to Y}\rangle = 1$.

□

2 Manual for the ION computational package

As described in the main text, we developed a computational package, ION (Inferring Oscillatory Networks), to infer networks of biological oscillators (Fig. 4A). Accompanying MATLAB code under a BSD-style license and examples are available at https://github.com/Mathbiomed/ION. Additionally, the code is available under a CC-BY 4.0 License at
https://doi.org/10.6084/m9.figshare.16431408.v1.

1. *Generate the data.csv file.* The first column of the data.csv file should be the time points at which the measurements were taken. The subsequent columns should be the data for each variable at the respective time points. If the data are stored in an excel spreadsheet, order the file as time points in column 1, data for variable 1 in column 2, data for variable 2 in column 3, etc., and then save it as a .csv file in the same directory as the code. Be sure that there are no variable names at the top of the data.csv file (see Input 1 in Fig. S1 for the structure of the data.csv file).

2. *Update the* `regulation_detection_inputs.m` *file.* Users need to specify the threshold used to infer regulation and the data interpolation method, in particular, either 'linear' or 'fourier'. The linear method will linearly interpolate the discrete data given in the data.csv file to create a continuous data set. The fourier method will fit a Fourier series to the discrete data to create a continuous data set.

   (a) *Update the threshold variable* (Input 2 in Fig. S1). The variable *threshold* determines the threshold for the $\overrightarrow{R}$ values accepted as interactions. For example, a threshold of 0.99 means that we relax the condition $\overrightarrow{R} = (\pm 1, -1)$ (Fig. 2A) up to ±0.99, i.e, we accept any interaction that satisfies both $|\langle R_{X\to Y}\rangle| > 0.99$ and $\langle R_{Y\to Y}\rangle < -0.99$. Based on our analysis in Fig. 3, we recommend a threshold of 0.9, which we use for inferring interactions given experimental data in Fig. 4.

   (b) *If linear method is chosen, update the variables 'supportlength' and 'modelorder'* (Input 2 in Fig. S1). They need to be chosen so that the function `movingslope.m` accurately estimates the derivatives at each time point of the data set [1]. The variable 'supportlength' determines the number of points used for the moving window average estimation of the derivative, so it should be at least 2 and no more than the number of data points. If 'supportlength' is odd, then the derivative is estimated using a central sliding window. If 'supportlength' is even, then the moving window will be moved backward by one element. For example, a 'supportlength' of 2 is equal to a difference quotient, computed for instance using the MATLAB function `diff`. Next, the variable 'modelorder' defines the order of the windowed model used to estimate the slope. If 'model order' is 1 or less than the 'supportlength'-1, then the model is linear or a regression, respectively. If 'modelorder' is equal to 'supportlength'-1, then the method will be a sliding Lagrange interpolant.

   (c) *If fourier method is chosen, update the 'num_fourier' variable* (Input 2 in Fig. S2). The variable 'num_fourier' determines the number of Fourier series coefficients and must be at least one and at most 8. For example, if 'num_fourier' = 1, then the algorithm fits the parameters a_0, a_1, b_1, and ω to the data for each variable based on the Fourier series of the form $F(t) = a_0 + a_1\cos(2\pi\cdot\omega\cdot t) + b_1\sin(2\pi\cdot\omega\cdot t)$. The algorithm fits the Fourier series with the specified order using a nonlinear least-squares method, specifically using the Levenberg-Marquardt method. To manually adjust the Fourier fitting method and options (e.g., the method, function tolerance, etc.), users need to update the function `createFit.m`.

3. *If linear method is chosen, create extrema value files for each variable and save them in the same directory as the scripts* (Input 3 in Fig. S1). If the fourier method is chosen, the interpolated time series has one maximum and minimum per cycle. However, if the linear method is chosen, local extrema can occur due to noise. To distinguish such local extrema from the global extrema per cycle, which is required to compute the reflection time (see Fig. 1B), users need to input extrema value files for each variable in the data set. The files should be saved as ek.csv where k corresponds to the variable. For example, the extrema value file for variable 1 (the first data column) should be saved as e1.csv. The extrema

value files should have time points in the first column and either 1 or -1 in the second column for a global maximum and a global minimum in that period at the time point, respectively. Users can generate these files by using the MATLAB function `findpeaks`. The function `findpeaks` returns all local minima and maxima, which users can use to decide the global minimum and maximum in each period.

4. *Run the* `main.m` *function.* The `main.m` function will read the inputs from the `regulation_detection_inputs.m` file and run the algorithm with the user-specified method. If the linear (fourier) method was chosen, the `main.m` function calls the `main_linear.m` (`main_fourier.m`) script to run the algorithm. Users will see several figures being generated as well as several output files generated in a new "Outputs" directory, which will be created within the same directory that the script is run:

   (a) If the linear method is chosen, a derivative.fig file that plots the estimated derivative values for all variables (Output 1 in Fig. S1).

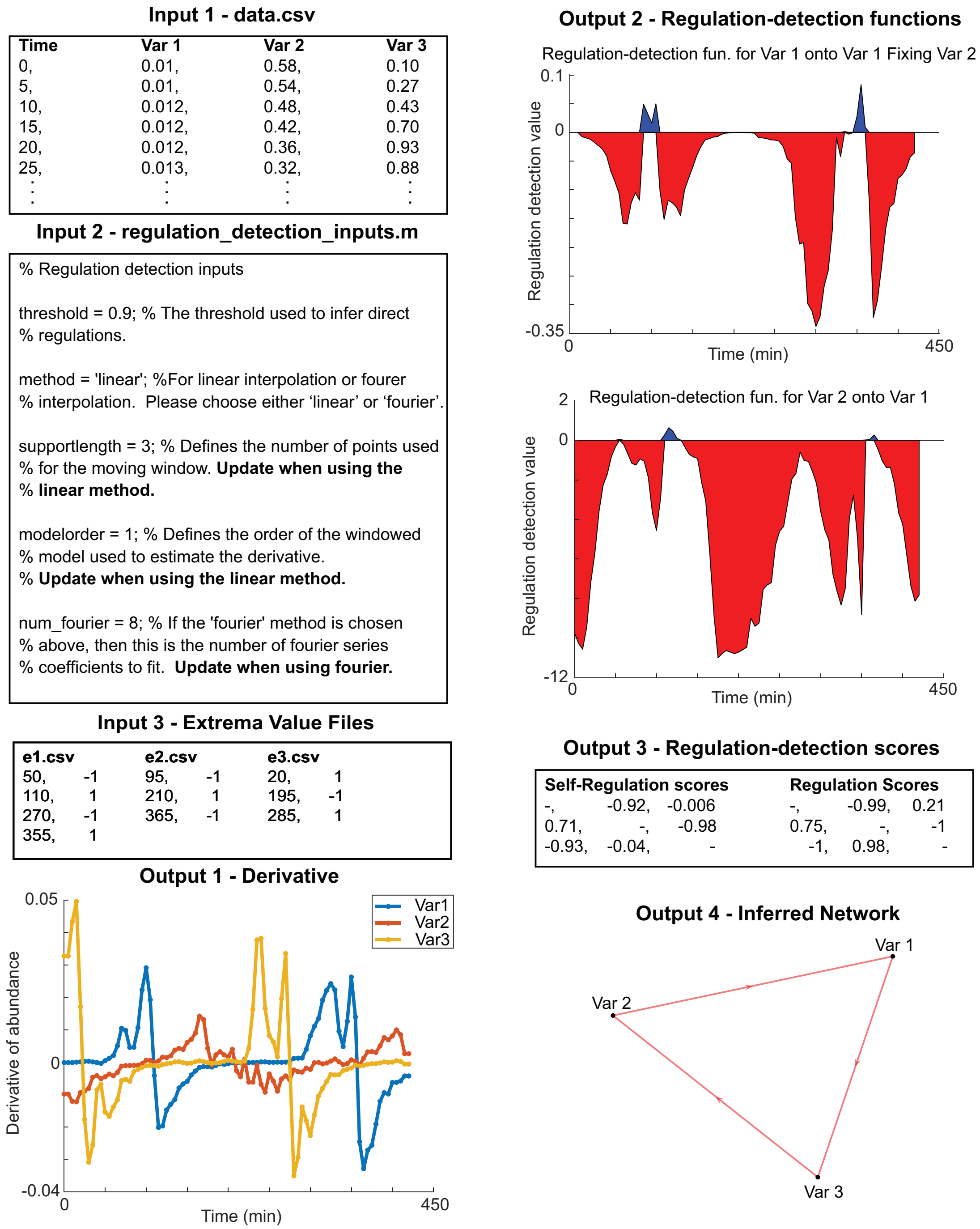
   (b) If the fourier method is chosen, files labeled Var$k$.fig, where k corresponds to the variable number, showing the fourier fits for each variable (Output 1 in Fig. S2).

   (c) Figures plotting the regulation-detection functions for each possible interaction (Output 2 in Fig. S1 and S2). For example, the file Reg_detect_i_onto_i_fix_j.fig plots the regulation-detection function $R^{t_{V_j}}_{V_i \to V_i}(t)$ (see Fig. 1C), where V_i is variable i. Similarly, the file Reg_detect_i_onto_j.fig plots the regulation-detection function $R^{t_{V_j}}_{V_i \to V_j}(t)$ (see Fig. 1C).

   (d) A .mat file with the regulation-detection scores for each possible self-regulation ('R_self' variable) and cross-regulation ('R_cross' variable) (Output 3 in Fig. S1 and S2).

   (e) A figure plotting the inferred network structure, inferred_network_graph.fig (Output 4 in Fig. S1 and S2). If R_self$(i, j) < -$threshold and —R_cross$(i, j)| >$ threshold, then the algorithm infers the interaction from Variable j to Variable i. Moreover, if R_cross(i, j) is positive (negative), then it is a positive (negative) regulation.

   (f) A Final_results.pdf document that reports each output listed above for reference.



Input 1 - data.csv

Time	Var 1	Var 2	Var 3
0,	0.01,	0.58,	0.10
5,	0.01,	0.54,	0.27
10,	0.012,	0.48,	0.43
15,	0.012,	0.42,	0.70
20,	0.012,	0.36,	0.93
25,	0.013,	0.32,	0.88
⋮	⋮	⋮	⋮

Input 2 - regulation_detection_inputs.m

```
% Regulation detection inputs

threshold = 0.9; % The threshold used to infer direct
% regulations.

method = 'linear'; %For linear interpolation or fourer
% interpolation.  Please choose either 'linear' or 'fourier'.

supportlength = 3; % Defines the number of points used
% for the moving window. Update when using the
% linear method.

modelorder = 1; % Defines the order of the windowed
% model used to estimate the derivative.
% Update when using the linear method.

num_fourier = 8; % If the 'fourier' method is chosen
% above, then this is the number of fourier series
% coefficients to fit.  Update when using fourier.
```

Input 3 - Extrema Value Files

e1.csv		e2.csv		e3.csv	
50,	-1	95,	-1	20,	1
110,	1	210,	1	195,	-1
270,	-1	365,	-1	285,	1
355,	1				

Output 1 - Derivative

Output 2 - Regulation-detection functions

Regulation-detection fun. for Var 1 onto Var 1 Fixing Var 2
0.1
0
-0.35
Regulation detection value
0
450
Time (min)
Regulation-detection fun. for Var 2 onto Var 1
2
0
-12
Regulation detection value
0
450
Time (min)

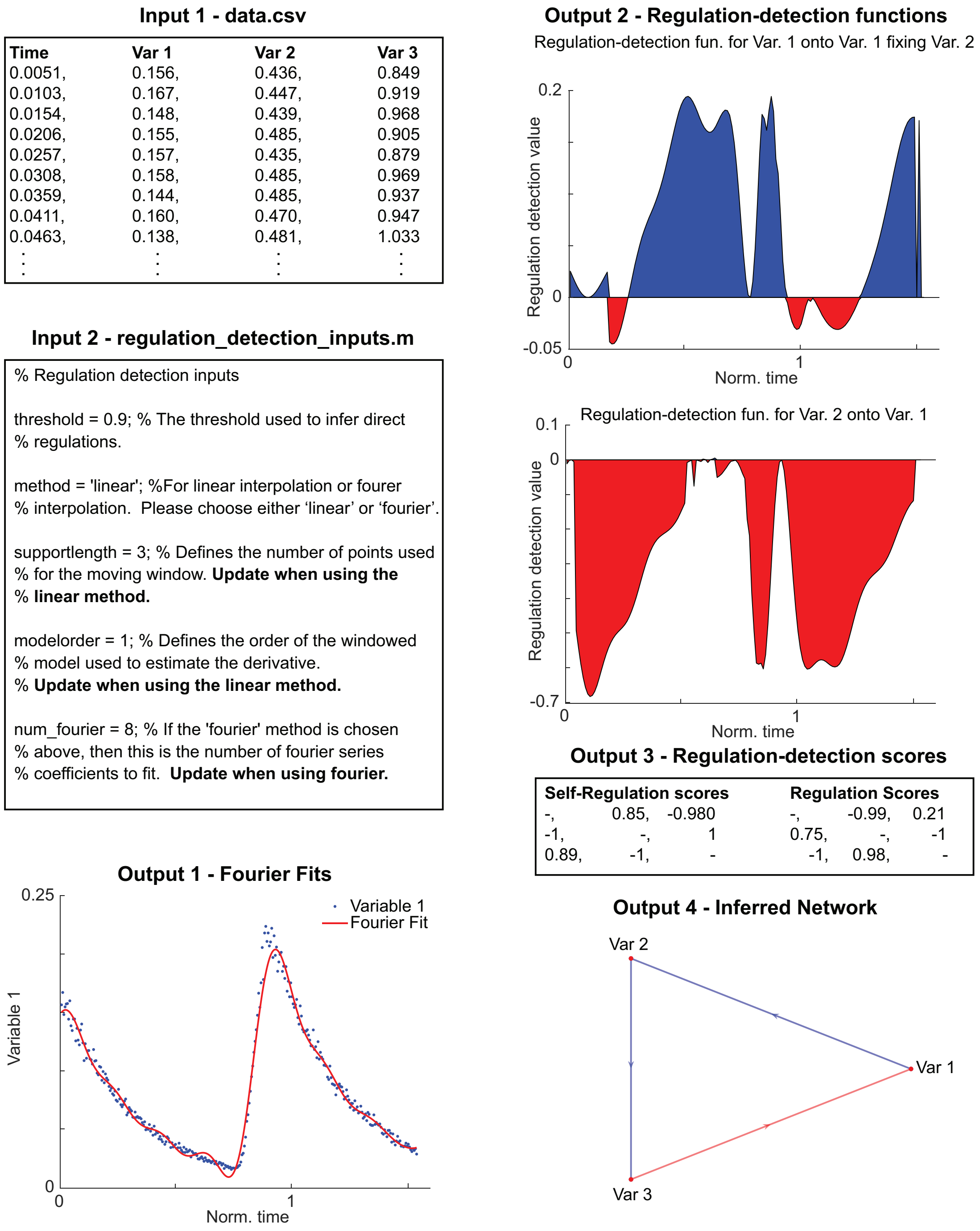
Output 3 - Regulation-detection scores

Self-Regulation scores			Regulation Scores		
-,	-0.92,	-0.006	-,	-0.99,	0.21
0.71,	-,	-0.98	0.75,	-,	-1
-0.93,	-0.04,	-	-1,	0.98,	-

Output 4 - Inferred Network

Figure S1: **Sample input and output files for the ION package if the linear method is chosen based on the experimental repressilator example (Fig. 4B)**. The data.csv (Input 1) file contains the time points in the first column and the data for each variable in subsequent columns, separated by commas. The regulation_detection_inputs.m file (Input 2) is already in the directory and should be updated according to the preferences of the user. If the user selects the 'linear' method, then the user must choose appropriate values for the 'supportlength' and

‘modelorder’ variables as well as generate extrema value files that list the time points at which the global maxima (1) and minima (-1) happen for each period (Input 3). The algorithm outputs the estimated derivatives for each variable (Output 1), plots of the regulation-detection functions (Output 2), the regulation-detection scores (Output 3), and the inferred network structure (Output 4).



Input 1 - data.csv

```
Time       Var 1     Var 2     Var 3
0.0051,    0.156,    0.436,    0.849
0.0103,    0.167,    0.447,    0.919
0.0154,    0.148,    0.439,    0.968
0.0206,    0.155,    0.485,    0.905
0.0257,    0.157,    0.435,    0.879
0.0308,    0.158,    0.485,    0.969
0.0359,    0.144,    0.485,    0.937
0.0411,    0.160,    0.470,    0.947
0.0463,    0.138,    0.481,    1.033
  ⋮          ⋮         ⋮         ⋮
```

Input 2 - regulation_detection_inputs.m

```
% Regulation detection inputs

threshold = 0.9; % The threshold used to infer direct
% regulations.

method = 'linear'; %For linear interpolation or fourer
% interpolation.  Please choose either 'linear' or 'fourier'.

supportlength = 3; % Defines the number of points used
% for the moving window. Update when using the
% linear method.

modelorder = 1; % Defines the order of the windowed
% model used to estimate the derivative.
% Update when using the linear method.

num_fourier = 8; % If the 'fourier' method is chosen
% above, then this is the number of fourier series
% coefficients to fit.  Update when using fourier.
```

Output 1 - Fourier Fits

Output 2 - Regulation-detection functions

Regulation-detection fun. for Var. 1 onto Var. 1 fixing Var. 2
0.2
0
-0.05
Regulation detection value
0
1
Norm. time
Regulation-detection fun. for Var. 2 onto Var. 1
0.1
0
-0.7
Regulation detection value
0
1
Norm. time

Output 3 - Regulation-detection scores

```
Self-Regulation scores              Regulation Scores
-,         0.85,   -0.980           -,        -0.99,    0.21
-1,          -,        1            0.75,       -,       -1
0.89,       -1,        -               -1,    0.98,       -
```

Output 4 - Inferred Network

Figure S2: **Sample input and output files for the ION package if the fourier method is chosen based on the Kim-Forger model with added noise (Fig. 2B)**. The data.csv (Input 1) file contains the time points in the first column and the data for each variable in subsequent columns, separated by commas. The regulation_detection_inputs.m file (Input 2) is already in the directory and should be updated according to the preferences of the user. If the user selects the 'fourier' method, then the algorithm fits a Fourier series to each variable with

the number of coefficients specified by the variable 'num_fourier' variable (Input 2). The algorithm outputs the Fourier fits for each variable (Output 1), plots of the regulation-detection functions (Output 2), the regulation-detection scores (Output 3), and the inferred network structure (Output 4).

3 Optimizing the threshold value

In Figure 4, we chose a threshold value of 0.9 based on the robustness analysis in Figure 3. However, depending on whether the goal is decreasing false-positive or negative predictions, one can adjust the threshold (i.e., increase or decrease) the threshold. For this, we define

$$\mathcal{P}_{X\to Y} = \begin{cases} \min|\overrightarrow{R}| & \text{if } R_{Y\to Y} < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{S4}$$

The value of $\mathcal{P}_{X\to Y}$ is between 0 and 1, and when it becomes closer to 1, the regulation $X \to Y$ is more likely to be true.

In Figure S3, we plot the values $\mathcal{P}_{X\to Y}$ for all possible regulations of the experimental repressilator example (Figure S3A), the mixed repressilator example (Figure S3B), and the estradiol example (Figure S3C). In each example, there is a significant drop off in the value $\mathcal{P}_{X\to Y}$ when moving from the correct to incorrect interactions. That is, the $\mathcal{P}_{X\to Y}$ values appear to exhibit a bi-modal distribution, where the values from the correct interactions are close to 1, and the values of the incorrect interactions taper off uniformly to 0. Thus, we expect that a natural choice of threshold can often be obtained from such bi-modal distributions, by plotting the distribution of $\mathcal{P}_{X\to Y}$ values.

Furthermore, setting a higher $\mathcal{P}_{X\to Y}$ threshold value decreases the number of false positives but increases the number of false negatives. On the other hand, lowering the $\mathcal{P}_{X\to Y}$ threshold decreases the number of false negatives but increases the number of false positives. Thus, users need choose the threshold value based on the level of precision or accuracy required from the specific application. Users are encouraged to examine the plot of the distribution of the $\mathcal{P}_{X\to Y}$ values provided by ION to determine an appropriate threshold for their specific application.

4 Description of the *in silico* models

4.1 Kim-Forger Model

The Kim-Forger model describes the core transcriptional-translational feedback loop in the molecular circadian clock (Fig. 2B) [2]. It is similar to the Goodwin model in that a protein product represses the transcription of its *mRNA*. However, the repression mechanism is not a

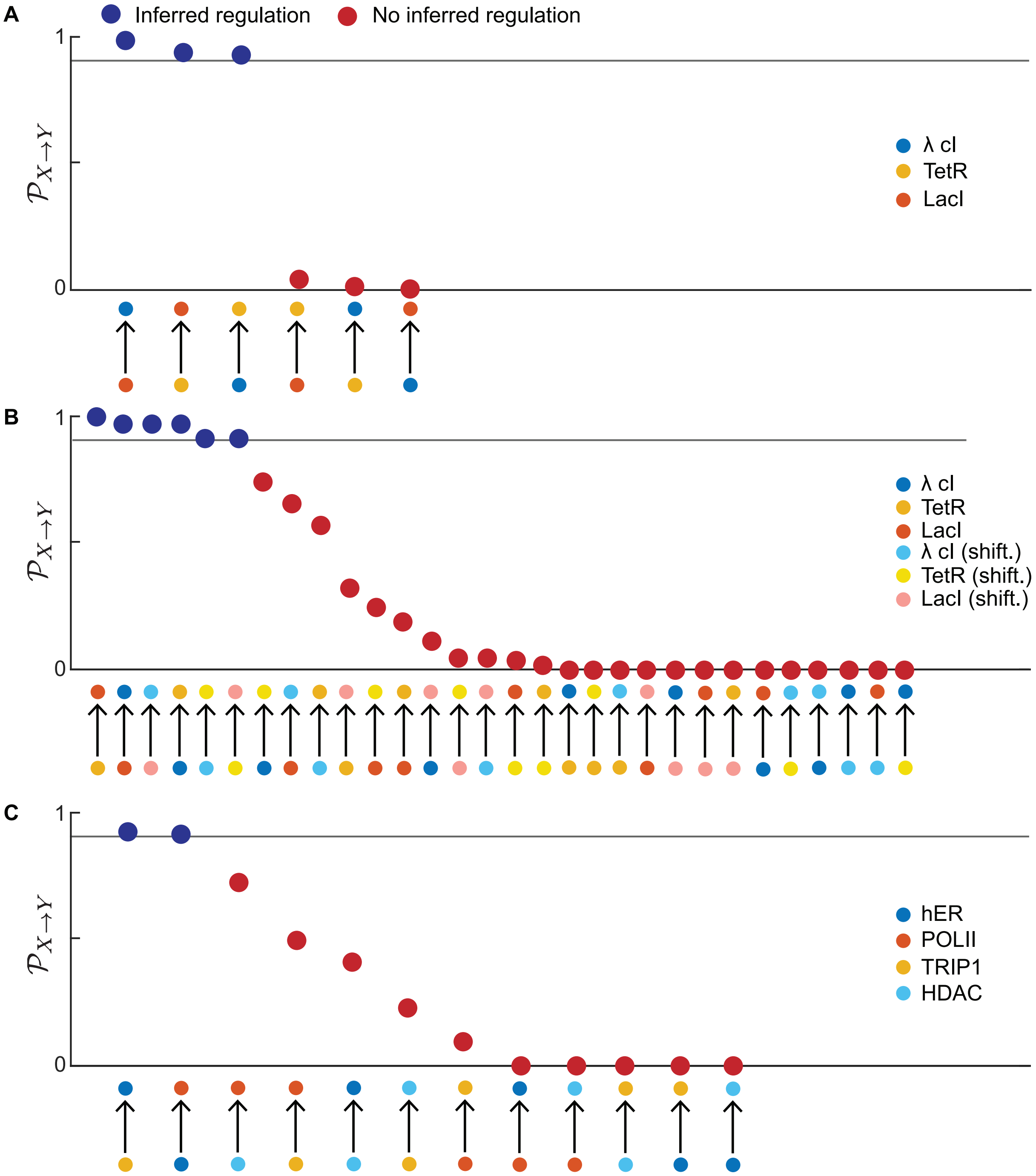


Figure S3: $\mathcal{P}_{X\to Y}$ **values for the three experimental data sets in Figure 4.** $\mathcal{P}_{X\to Y}$ (Eqn. (S4)) of all possible interactions is plotted for (A) the repressilator example (Figure 4B), (B) the mixed repressilator example (Figure 4C), and (C) the estradiol example (Figure 4C). Here $\mathcal{P}_{X\to Y} = 0.9$ (solid line) is used as the threshold for inference.

Hill-type repression mechanism as in the Goodwin oscillator, but rather a protein sequestration mechanism where tight binding of the activator and repressor sequesters the activator protein, which in turn represses its activity. The model is given by the following system of ODEs.

$$\begin{aligned}
\frac{\mathrm{d}M}{\mathrm{d}t} &= k_1 f(P) - k_2 M \\
\frac{\mathrm{d}P_C}{\mathrm{d}t} &= k_1 M - k_3 P_C \\
\frac{\mathrm{d}P}{\mathrm{d}t} &= k_1 P_C - k_4 P \\
f(z) &= \frac{A_t - K_d - z + \sqrt{(A_t - K_d - P)^2 - 4A_t K_d}}{2A_t}.
\end{aligned}$$

Here, the variable M is the repressor mRNA concentration; P_C is the cytosolic repressor protein concentration; P is the nuclear repressor protein concentration. The parameter A_t is the total activator concentration, and K_d is the dissociation constant between the activator and repressor (closer to 0 means tighter binding). As in [3], we simulate the Kim-Forger model with the parameters $k_1 = 1$, $k_2 = 0.16$, $k_3 = 0.29$, $k_4 = 0.3$, $A_t = 0.6$, and $K_d = 10^{-5}$.

4.2 Frzilator

The *frzilator* model consists of a negative feedback loop that models the oscillations in a cascade of covalent modifications in *Myxococcus xanthus* [4] (Fig. 2C). The mathematical model uses Michaelis-Menten dynamics to model the negative feedback and is given by the equations

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}t} &= \phi\left(\frac{1-f}{0.01 + (1-f)}\right) - d_f\left(\frac{f}{0.005 + f}\right)e \\
\frac{\mathrm{d}c}{\mathrm{d}t} &= k_c\left(\frac{1-c}{0.005 + (1-c)}\right)f - d_c\left(\frac{c}{0.005 + c}\right) \\
\frac{\mathrm{d}c}{\mathrm{d}t} &= k_e\left(\frac{1-e}{0.005 + 1 - e}\right) - d_e\left(\frac{e}{0.005 + e}\right).
\end{aligned}$$

As in [3], we simulate the system with the nominal parameter values $\phi = 0.08$, $k_c = 4$, $k_e = 4$, $d_f = 1$, $d_c = 2$, and $d_e = 2$.

4.3 Goodwin Oscillator

The Goodwin model describes the action of a protein product repressing its *mRNA* [5] (Fig. 2D). The transcriptional regulation is described by a Hill function, where the Hill coefficient is an upper bound for the number of repressor proteins that bind to the promoter [6]. A high Hill coefficient is required for the system to generate sustained oscillations. We simulated the

Goodwin oscillator from the following system of ODEs.

$$
\begin{aligned}
\frac{\mathrm{d}M}{\mathrm{d}t} &= \frac{1}{1+P_3^{10}} - 0.4M \\
\frac{\mathrm{d}P_1}{\mathrm{d}t} &= M - 0.4P_1 \\
\frac{\mathrm{d}P_2}{\mathrm{d}t} &= P_1 - 0.4P_2 \\
\frac{\mathrm{d}P_3}{\mathrm{d}t} &= P_2 - 0.4P_3
\end{aligned}
$$

4.4 Kim-Forger Model with Outputs

We augment the original Kim-Forger model (Section 4.1) so that each component promotes the production of one output, e.g., M positively regulates X_M (Fig. 2F). The system of ODEs is given below.

$$
\begin{aligned}
\frac{\mathrm{d}M}{\mathrm{d}t} &= k_1 f(P) - k_2 M \\
\frac{\mathrm{d}P_C}{\mathrm{d}t} &= k_1 M - k_3 P_C \\
\frac{\mathrm{d}P}{\mathrm{d}t} &= k_1 P_C - k_4 P \\
\frac{\mathrm{d}X_M}{\mathrm{d}t} &= k_5 \frac{M^{n_1}}{1+M^{n_1}} - k_6 X_M \\
\frac{\mathrm{d}X_{P_C}}{\mathrm{d}t} &= k_7 \frac{P_C^{n_2}}{1+P_C^{n_2}} - k_8 X_{P_C} \\
\frac{\mathrm{d}X_P}{\mathrm{d}t} &= k_9 \frac{P^{n_3}}{1+P^{n_3}} - k_{10} X_P \\
f(z) &= \frac{A_t - K_d - z + \sqrt{(A_t - K_d - P)^2 - 4A_t K_d}}{2A_t}.
\end{aligned}
$$

Here, the parameters k_1 through k_4 are the same as in Section 4.1. We simulate the model with the parameters $k_5 = 0.5$, $k_6 = 0.59$, $k_7 = 0.23$, $k_8 = 0.98$, $k_9 = 0.08$ and $k_{10} = 0.63$ and Hill coefficients $n_1 = 10$, $n_2 = 15$, and $n_3 = 20$.

4.5 Repressilator

The repressilator is a synthetic feedback loop that consists of three genes and three proteins where the mRNAs translate to the respective proteins, which in turn repress the transcription of the next mRNA (Fig. 2G) [7]. The model is given by the following ODEs.

$$
\begin{aligned}
\frac{\mathrm{d}M_1}{\mathrm{d}t} &= \frac{\alpha}{1+P_3^n} - M_1 \\
\frac{\mathrm{d}P_1}{\mathrm{d}t} &= \beta(M_1 - P_1) \\
\frac{\mathrm{d}M_2}{\mathrm{d}t} &= \frac{\alpha}{1+P_1^n} - M_2 \\
\frac{\mathrm{d}P_2}{\mathrm{d}t} &= \beta(M_2 - P_2) \\
\frac{\mathrm{d}M_3}{\mathrm{d}t} &= \frac{\alpha}{1+P_2^n} - M_3 \\
\frac{\mathrm{d}P_3}{\mathrm{d}t} &= \beta(M_3 - P_3),
\end{aligned}
$$

where $\alpha = 7$, $\beta = 3$, and $n = 10$.

5 Complete results of *in silico* and experimental network inference (Figs. 2 and 4)

In Tables S1-S5, we report the complete results of our network inference procedure applied to the *in silico* examples in Fig. 2. In Tables S6-S8, we report the complete results of our network inference procedure applied to the experimental data sets (Fig. 4). The inferred interactions are the interactions that pass Rules 1-3 (Fig. 2A) with a threshold value of 0.90. Each table cell is $(\langle R_{X\to Y}\rangle, \langle R_{Y\to Y}\rangle)$ where X is the column variable and Y is the row variable.

	f	c	e
f	–	$(-1, 1)$	$(-1, -1)$
c	$(1, -1)$	–	$(-1, 0.99)$
e	$(1, -0.71)$	$(1, -1)$	–

Table S1: Calculated regulation-detection scores for the Frzilator oscillator example (Fig. 2C).

	M	P_1	P_2	P_3
M	–	$(-1, 0.98)$	$(-1, 0.61)$	$(-1, -1)$
P_1	$(1, -1)$	–	$(-1, 0.92)$	$(-1, 0.21)$
P_2	$(1, -0.30)$	$(1, -1)$	–	$(-1, 0.64)$
P_3	$(1, 0.06)$	$(1, -0.23)$	$(1, -1)$	–

Table S2: Calculated regulation-detection scores for the Goodwin oscillator example (Fig. 2D).

	M_1	P_C	P	M_2	P_1	P_2	P_3
M_1	–	$(-1, 0.68)$	$(-1, -1)$	$(-0.90, -0.56)$	$(-0.17, -0.39)$	$(0.87, -0.22)$	$(1, -0.23)$
P_C	$(1, -1)$	–	$(-1, 1)$	$(-1, -0.84)$	$(-0.90, -0.54)$	$(0.11, 0.04)$	$(0.98, 0.81)$
P	$(1, 0.76)$	$(1, -1)$	–	$(-1, -0.18)$	$(-1, -0.87)$	$(-0.97, -0.92)$	$(0.03, -0.15)$
M_2	$(0.75, -0.25)$	$(0.99, -0.17)$	$(1, -0.23)$	–	$(-1, 0.91)$	$(-1, -0.02)$	$(-1, -1)$
P_1	$(-0.28, -0.26)$	$(0.82, 0.18)$	$(1, 0.43)$	$(1, -1)$	–	$(-1, 1)$	$(-1, 0.36)$
P_2	$(-0.92, -0.85)$	$(-0.22, -0.54)$	$(0.97, 0.68)$	$(1, -0.42)$	$(1, -1)$	–	$(-1, 1)$
P_3	$(-1, -0.97)$	$(-0.97, -0.97)$	$(-0.03, -0.55)$	$(1, 0.76)$	$(1, -0.68)$	$(1, -1)$	–

Table S3: Calculated regulation-detection scores for the Kim-Forger, Goodwin independent cycle example (Fig. 2E).

	M_1	P_C	P	X_M	X_{P_C}	X_P
M_1	–	$(-1, 0.65)$	$(-1, -1)$	$(-1, 1)$	$(-1, -0.75)$	$(-0.99, -0.91)$
P_C	$(1, -1)$	–	$(-1, 1)$	$(1, -0.92)$	$(-1, 0.87)$	$(-1, 0.51)$
P	$(1, 0.87)$	$(1, -1)$	–	$(1, -0.87)$	$(1, -0.97)$	$(-1, 1)$
X_M	$(1, -1)$	$(-1, 0.45)$	$(-1, -0.40)$	–	$(-1, -0.05)$	$(-1, -0.97)$
X_{P_C}	$(1, -0.03)$	$(1, -1)$	$(-1, 0.69)$	$(1, -0.08)$	–	$(-1, 0.35)$
X_P	$(0.98, 0.31)$	$(1, 0.12)$	$(1, -1)$	$(0.99, 0.21)$	$(1, -0.51)$	–

Table S4: Calculated regulation-detection scores for the Kim-Forger with outputs example (Fig. 2F).

	M_1	P_1	M_2	P_2	M_3	P_3
M_1	–	$(-1, 1)$	$(1, 0.70)$	$(1, 0.17)$	$(-1, -0.85)$	$(-1, -1)$
P_1	$(1, -1)$	–	$(1, 0.98)$	$(1, 0.90)$	$(-1, -0.23)$	$(-1, -1)$
M_2	$(-1, -0.82)$	$(-1, -1)$	–	$(-1, 1)$	$(1, 0.71)$	(0.22)
P_2	$(-1, -0.15)$	$(-1, -1)$	$(1, -1)$	–	$(1, 0.98)$	$(1, 0.90)$
M_3	$(1, 0.79)$	$(1, 0.33)$	$(-1, -0.80)$	$(-1, -1)$	–	$(-1, 1)$
P_3	$(1, 0.99)$	$(1, 0.93)$	$(-1, -0.13)$	$(-1, -1)$	$(1, -1)$	–

Table S5: Calculated regulation-detection scores for the repressilator example (Fig. 2G).

	P_1	P_2	P_3
P_1	–	$(-0.99, -0.92)$	$(0.21, -0.006)$
P_2	$(0.75, 0.71)$	–	$(-1, -0.98)$
P_3	$(-1, -0.93)$	$(0.98, -0.04)$	–

Table S6: Calculated regulation-detection scores for the experimental repressilator example (Fig. 4B).

	P_1^1	P_2^1	P_3^1	P_1^2	P_2^2	P_3^2
P_1^1	–	$(-1, -0.96)$	$(0.56, -0.18)$	$(-0.83, -0.01)$	$(0.97, -0.57)$	$(-1, 0.37)$
P_2^1	$(0.83, 0.88)$	–	$(-1, -0.97)$	$(-0.65, 0.99)$	$(-0.19, 0.18)$	$(0.96, 0.14)$
P_3^1	$(-1, -1)$	$(0.96, 0.02)$	–	$(1, -0.03)$	$(-1, 0.16)$	$(-0.96, 0.05)$
P_1^2	$(-0.69, 0.37)$	$(0.93, -0.74)$	$(-1, -0.24)$	–	$(-0.99, -0.91)$	$(-0.04, -0.09)$
P_2^2	$(-0.97, 0.98)$	$(0.15, 0.38)$	$(1, -0.65)$	$(0.76, 0.61)$	–	$(-1, -0.96)$
P_3^2	$(1, -0.31)$	$(-1, -0.11)$	$(-0.67, 0.23)$	$(-1, -0.91)$	$(0.97, -0.04)$	–

Table S7: Calculated regulation-detection scores for the mixed repressilator example (Fig. 4C).

	hER	$POLII$	$TRIP1$	$HDAC$
hER	–	$(-0.78, 1)$	$(-0.94, -0.92)$	$(-0.41, -0.92)$
$POLII$	$(0.91, -0.99)$	–	$(-0.99, -0.49)$	$(-0.97, -0.72)$
$TRIP1$	$(0.87, 0.58)$	$(0.93, -0.10)$	–	$(-0.73, 0.66)$
$HDAC$	$(0.80, 1)$	$(0.92, 0.86)$	$(0.88, -0.23)$	–

Table S8: Calculated regulation-detection scores for the estradiol example (Fig. 4D).

References

[1] J. D'Errico, Movingslope (2020).

[2] J. K. Kim, D. B. Forger, *Molecular Systems Biology* **8**, 630 (2012).

[3] V. Bokka, *IET Systems Biology* **12**, 190 (2018).

[4] O. A. Igoshin, A. Goldbeter, D. Kaiser, G. Oster, *Proceedings of the National Academy of Sciences* **101**, 15760 (2004).

[5] B. C. Goodwin, *Advanced Enzyme Regulation* **3**, 425 (1965).

[6] Santillán, M., *Math. Model. Nat. Phenom.* **3**, 85 (2008).

[7] M. B. Elowitz, S. Leibler, *Nature* **403**, 335 (2000).